Tommy Las Z23517623 Tommy-Las

April 10th 2022 [Github Repository](https://github.com/COT4930-Spring-2022-Fullstack-Web/hw6-Tommy-Las)

Full Stack Web Development COT4930

Homework #6 – Part 2

Part 1:

For the first part, I had to copy the student server from HW5 into a folder called ‘backend’ in HW6. I used the same student server from HW5.

Part 2:

First, I’m going to mention that most of the frontend was created using Bootstrap. I installed a Bootstrap package using npm install react-bootstrap, that has bootstrap components. For example, if you want to add a bootstrap container for example, you would use <Container> (here you add other elements) </Container>. Bootrsap with react is very easy to use.

Also, to send the HTTP Requests, for this homework I used the axios which is a is a promise-based HTTP Client for node.js. It is way easier to use than AJAX or Fetch in my opinion. You first install it using npm install axios. Then you can use either axios.get() or axios.put() or axios.post() or axios.delete() for each request. In the first parameter you include the address to send the request, in this case ‘http://localhost:5678/students’, the second parameter you include any parameters that you want to send if needed. Then you must use .then and .catch. In .then you include what you want to perform in case the request is successful. In .catch you include what you want to do in case of an error.

First, you need to start the server using node studentserver.js in the backend folder. Then I run npm start in the frontend folder.

This is my home page, instead of having a button or a list that sends you to each page, I implemented a navigation bar on top of the screen, which makes it more user friendly. The navigation bar will appear in all pages, so you can have access to all pages. The navigation bar is implemented using the React Route, which I think it was the hardest React material to learn for me. I implemented the Routes in the App.js. For example, the route with path ‘add’ will take you to the component <AddStudent />. I did this with all the other components.

Text

Description automatically generated

Homepage:

Graphical user interface, text, application, email

Description automatically generated

Add student to the database:

This is my page to add a student to the database. The form was made using Bootstrap. On the bottom it shows either an error message or a success message. In this case is a success message and you can see that all the input fields were cleared since the request was successful. Each input field is controlled using the useState hook. The message on the bottom is also controlled using the useState hook.

Graphical user interface, text, application

Description automatically generated

In this case, the user is already in the system, so the inputs field were not cleared. Also, it shows the error in the bottom.

Graphical user interface, application

Description automatically generated

Delete student from the database:

I used the useState hook to handle the input of the student ID. I used axios.delete(‘http://localhost:5678/students’ + id).

Graphical user interface, application

Description automatically generated

Update student’s information:

I used the useState hook to control all input fields. I used axios.put(‘http://localhost:5678/students’ + id, params) where params is an object with the values needed to be changed.

Graphical user interface, application

Description automatically generated

Search student by id:

The value of the input field is controlled using the useState hook.

The table on the bottom to show the student was made using bootstrap. I created an empty table first, but it won’t show nothing until the table variable is updated with the student that was searched. If no student is found, it will show an error message, and no table is shown. When a student is found, I use setTable to set the variable to a table row:

**Text

Description automatically generated**

A picture containing graphical user interface

Description automatically generated

Search student by full name:

I used the useState hook to control the first name and last name input. The user information is show on a table the same way as searching a student by student id. On the first place, I wanted to have only one page for searching with id and searching by name, but I couldn’t come up with a way to have both components (search by id, search by name) in one single component.

Graphical user interface

Description automatically generated

Displaying all students:

The table with all the students loads as soon as you open the page. This is done using the useEffect hook, which is going to run the sendRequest function on the first render. The empty brackets mean to run it on the first render

Text

Description automatically generated

I created an empty table first that will hold all the values. If the database is empty, it will show an empty table. If the request is successful, the server is going to respond with an array of all the student objects. I use the .map method to convert all of the elements the student array into table row elements. Then using the setTable() I would change the state to the array and it would display all students in the table.

Text

Description automatically generated

Graphical user interface, table

Description automatically generated